REMARKS

I. Summary of the Office Action and this Reply

Claims 1, 3-13, 16 and 18-32 are pending in the application. Claims 1, 3-13 and 31 stand rejected under 35 U.S.C. §103(a) as being obvious over U.S. Patent No. 6,530,075 to Beadle ("Beadle") in view of U.S. Patent No. 7,222,218 to Dutt et al. ("Dutt"). Claims 16, 18-30 and 32 under 35 U.S.C. §103(a) stand rejected as obvious over Beadle in view of Dutt and U.S. Patent Application No. 2004/0221272 to Wu. The Examiner objected to claims 31 and 32.

In this Reply, claims 1, 4, 5, 9, 10, 11, 16, 19, 20 and 23 are amended. No new matter is added; support for the amended claims can be found, *inter alia*, in the application at page 1, line 9 – page 2, line 2; page 5, lines 13-19; page 9, lines 5-9; page 9, line 32 - page 10, line 16; page 10, lines 21-27; page 12, lines 12-22; page 13, lines 4-9; page 13, line 30 – page 24. Claims 31 and 32 have been canceled, obviating the objections on pages 2-3 of the Action.

II. Brief Discussion of the Claimed Invention

One aspect of the claimed invention relates to annotating p-code methods with specific priority level hints, and enabling preferential processing of the p-code methods based upon the priority levels assigned. By assigning priority levels, a memory-constrained target environment that has insufficient memory to JIT compile all p-code will devote the available memory to storing JIT compilation of portions of the p-code according to the order of priority assigned to those portions of the p-code. Thus, in a simplified example, if only one p-code method can be compiled (or stored in a cache) in view of limited resources, those limited resources will be devoted to compiling (or storing) the one p-code method having been annotated to have the highest priority; the limited resources will not be devoted to compiling (or storing) a lower

prioritized p-code method before a higher prioritized p-code method. See Summary.

III. Brief Discussion of the Selected Art

A. Beadle

Beadle discloses just-in-time (JIT)/Compiler Java language extensions to enable field performance and serviceability. Specifically, Beadle provides a programmer with keyword extensions of the Java language to indicate which Java objects, classes, methods or code sections are to be just-in-time compiled. Thus, Beadle teaches that certain Java objects may be marked to be JIT compiled.

B. Dutt

Dutt discloses a marking API for marking code sections of sequential code to indicate points at which multi-threading may be introduced to execute blocks of code in a concurrent fashion. A scheduler may schedule the marked blocks of concurrent code for concurrent execution, by multi-threaded processing. The scheduler may schedule blocks to execute concurrently in an ordered manner, e.g. to schedule longer run time blocks ahead of shorter run time blocks. See Abstract; col. 5, line 42 – col. 6, line 52; col. 12, lines 39 – 56; col. 19, lines 39-59. Accordingly, Dutt discloses a system and/or method for dividing sequential code into blocks of code that can be executed concurrently.

IV. Response to 103 Rejections

It is well-established that for a proper rejection under section 103, all claim limitations must be taught or suggested by the prior art.

The present invention specifically teaches enabling preferential processing based upon priority levels assigned to p-code methods in a p-code file. Neither Beadle nor Dutt discloses prioritizing p-code methods and enabling preferential processing of the p-code methods based on hierarchically-ordered priority levels as claimed. Thus, Beadle and Dutt, alone and in combination, fail to teach or suggest all claim limitations. For at least these reasons, Applicants submit that the cited art does not provide a basis for an obviousness rejection. Reconsideration and withdrawal of the rejection of claims 1, 3-13, 16 and 18-30 are requested respectfully.

A. Beadle Does Not Teach Or Suggest Priority-Based Annotations Enabling Preferential Processing Of Annotated Methods In A Hierarchical Order

Beadle teaches that certain Java objects may be marked to be compiled. However, Beadle does not teach assigning of priority-related information to code sections to be compiled. Beadle does not teach or suggest providing any priority-based annotations, or providing annotations enabling preferential processing of annotated methods in a hierarchical order. Instead, Beadle teaches only providing markers to distinguish between portions to be compiled, and portions that are not to be compiled.

Accordingly, Beadle fails to teach or suggest "annotating said identified p-code methods to be compiled, said annotating associating a respective priority level hint with each p-code method to be compiled, said priority level hints enabling preferential processing of said p-code methods in a hierarchical manner corresponding to a hierarchical order of said priority level hints" as required by claim 1.

Likewise, Beadle fails to teach or suggest " identifying one or more p-code methods within said p-code file that are annotated with a respective priority indicative annotation" and "storing said compiled p-code methods in a cache . . . , said compiled p-code methods being preferentially

retained in said cache in a hierarchical manner corresponding to a hierarchical order of their respective priority indicative annotations" as required by claim 16.

Thus, Beadle fails to teach or suggest all claim limitations.

B. Beadle And Dutt In Combination Fail To Teach Or Suggest Priority-Based Annotations Enabling Preferential Processing Of Annotated Methods In A Hierarchical Order

As discussed above, Beadle fails to teach of suggest priority-based annotations enabling preferential processing of annotated methods in a hierarchical order. Dutt discloses a system and/or method for dividing sequential code into blocks of code that can be executed concurrently. However, Dutt fails to teach or suggest "annotating said identified p-code methods to be compiled, said annotating associating a respective priority level hint with each p-code method to be compiled, said priority level hints enabling preferential processing of said p-code methods in a hierarchical manner corresponding to a hierarchical order of said priority level hints" as required by claim 1, or "identifying one or more p-code methods within said p-code file that are annotated with a respective priority indicative annotation" and "storing said compiled p-code methods in a cache . . . , said compiled p-code methods being preferentially retained in said cache in a hierarchical manner corresponding to a hierarchical order of their respective priority indicative annotations" as required by claim 16.

Dutt's disclosure of execution of code segments in a particular sequence is unrelated to the claimed invention. The claimed invention does not relate to code compilation in a particular order or sequence. Rather, the claimed invention relates generally to (1) processing of p-code methods annotated with specific priority levels, and processing the annotated methods in a hierarchical order corresponding to a hierarchical order of their priority levels, or (2)

preferentially retaining compiled p-code methods in a hierarchical manner corresponding to a hierarchical order of the priority level annotations, as recited with greater specificity in claims 1 and 16. By way of example, all (or many) p-code methods may be processed sequentially as sequential code, regardless of their respective priority annotations, and the compiled p-code methods may be retained in a limited-capacity cache according to a priority level, e.g. such that an earlier-compiled lower-priority p-code method is deleted from a cache and replaced with a later-compiled higher-priority p-code method, which is then retained in the cache despite subsequent processing of lower-priority p-code methods. See application, page 9, lines 5-27; page 12, lines 12-22.

Further, the Examiner asserts that the references would be combined for a specific purpose, namely, as stated on page 5 of the Action, "to schedule a plurality of blocks of concurrent code for multi-threaded execution as suggested by Dutt." This asserted reason for the combination is flawed. The claimed invention does not relate to multi-threading or concurrent execution of blocks of code. Instead, the claimed invention relates to preferential processing (such as retention within a limited-capacity cache), with preference being given as a function of a hierarchy of assigned priority levels reflected in priority level hints; the claimed invention does not relate to an assigned order or sequence in which portions of the code are to be processed. Stated differently, the "order" in Dutt may relate to an order of execution of code segments selected from sequential code; the "order" in the context of the present invention relates to an order of priority levels within a hierarchy, and does not relate to an order of processing of code segments.

Accordingly, there is no teaching or suggestion, in Beadle or Dutt, of the claimed invention. Further, neither Beadle nor Dutt provides a reason for combining Beadle and Dutt or

modifying the teachings of Beadle to arrive at the claimed invention. Further still, the claimed invention is not otherwise rendered obvious by the cited art. For at least these reasons, reconsideration and withdrawal of the rejections of claims 1 and 16 are requested respectfully.

Claims 3-13 depend from claim 1 and are likewise patentable. Claims 18-30 depend from claim 16 and are likewise patentable. For at least these reasons, reconsideration and withdrawal of the rejection of claims 2-13 and 18-30 are requested respectfully.

CONCLUSION

In view of the foregoing amendments and remarks, Applicants believe claims 1, 3-13, 16 and 18-30 to be patentable and the application in condition for allowance, and request respectfully issuance of a Notice of Allowance. If any issues remain, the undersigned requests a telephone interview prior to the issuance of an action.

Respectfully submitted,
Jeffrey Wannamaker et al.
by:

Date: April 4, 2008

/gregory s. bernabeo/
Gregory S. Bernabeo
Reg. No. 44,032

Synnestvedt & Lechner LLP
1101 Market Street, Suite 2600
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189